SPRAWOZDANIE 3

Zajęcia: Grafika komputerowa

Prowadzący:
prof. dr hab. Vasyl Martsenyuk

Laboratorium: Grafika Komputerowa 04.03.2020
Temat: Modelowanie hierarchiczne w grafice 2D

Sebastian Pierog
Informatyka I stopień, stacjonarne, 4 semestr, Gr.1b

Polecenie:

Opracować scenę hierarchiczną zgodnie z obrazem używając zamiast kół wielokąty obracające się (animacja!) według wariantu. Opracowanie powinno być w jednym z języków: Java lub JavaScript,

na dwa sposoby:

(a) używając hierarchię funkcje (sposób subroutine)

(b) tworząc graf sceny (sposób obiektowy). W tym celu proponuję do pobrania odpowiedni pliki

Kod źródłowy:
Zadanie1. Hierarchia

```
    private void drawWorld(Graphics2D g2) {

        // TODO: Draw the content of the scene.
        // wywolywanie funkcji
        nShape1(g2);
        nShape2(g2);
        nShape3(g2);
        nShape4(g2);
        nShape5(g2);
        nShape6(g2);
        bar1(g2);
        bar2(g2);
        bar3(g2);
        triangle1(g2);
        triangle2(g2);
        triangle3(g2);
    } // end drawWorld()
```

```
private void nShape1(Graphics2D g2) { // (DELETE THIS EXAMPLE)
    AffineTransform saveTransform = g2.getTransform();  // (It might be neces
    Color saveColor = g2.getColor();
    g2.setColor( Color.black );
    g2.translate( tx: -3.45, ty: 1.50);
    g2.rotate( Math.toRadians( frameNumber*0.75 ));
    g2.scale(  sx: 0.5, sy: 0.5 );
    nShape(g2);
    g2.setColor(saveColor);
    g2.setTransform(saveTransform);
}
private void nShape2(Graphics2D g2) { // (DELETE THIS EXAMPLE)
    AffineTransform saveTransform = g2.getTransform();  // (It might be neces
    Color saveColor = g2.getColor();
    g2.setColor( Color.black );
    g2.translate( tx: -1.55, ty: 1);
    g2.rotate( Math.toRadians( frameNumber*0.75 ));
    g2.scale(  sx: 0.5, sy: 0.5);
    nShape(g2);
    g2.setColor(saveColor);
    g2.setTransform(saveTransform);
}

private void nShape3(Graphics2D g2) { // (DELETE THIS EXAMPLE)
    AffineTransform saveTransform = g2.getTransform();  // (It might be neces
    Color saveColor = g2.getColor();
    g2.setColor( Color.black );
    g2.translate( tx: -1.15, ty: -0.20);
    g2.rotate( Math.toRadians( frameNumber*0.75 ));
    g2.scale(  sx: 0.7, sy: 0.7);
    nShape(g2);
    g2.setColor(saveColor);
    g2.setTransform(saveTransform);
}
```

```java
private void nShape4(Graphics2D g2) { // (DELETE THIS EXAMPLE)
    AffineTransform saveTransform = g2.getTransform();  // (It might be necessary to sa
    Color saveColor = g2.getColor();
    g2.setColor( Color.black );
    g2.translate( tx: 1.15, ty: -0.80);
    g2.rotate( Math.toRadians( frameNumber*0.75 ));
    g2.scale(  sx: 0.7, sy: 0.7 );
    nShape(g2);
    g2.setColor(saveColor);
    g2.setTransform(saveTransform);
}
private void nShape5(Graphics2D g2) { // (DELETE THIS EXAMPLE)
    AffineTransform saveTransform = g2.getTransform();  // (It might be necessary to sa
    Color saveColor = g2.getColor();
    g2.setColor( Color.black );
    g2.translate( tx: 1.8, ty: 1.5);
    g2.rotate( Math.toRadians( frameNumber*0.75 ));
    g2.scale(  sx: 0.3, sy: 0.3 );
    nShape(g2);
    g2.setColor(saveColor);
    g2.setTransform(saveTransform);
}
private void nShape6(Graphics2D g2) { // (DELETE THIS EXAMPLE)
    AffineTransform saveTransform = g2.getTransform();  // (It might be necessary to sa
    Color saveColor = g2.getColor();
    g2.setColor( Color.black );
    g2.translate( tx: 3.2, ty: 1.1);
    g2.rotate( Math.toRadians( frameNumber*0.75 ));
    g2.scale(  sx: 0.3, sy: 0.3 );
    nShape(g2);
    g2.setColor(saveColor);
    g2.setTransform(saveTransform);
}
```

```java
private void bar1(Graphics2D g2) { // (DELETE THIS EXAMPLE)
    AffineTransform saveTransform = g2.getTransform();  // (It might be ne
    Color saveColor = g2.getColor();
    g2.setColor( Color.red);
    g2.translate( tx: -2.5, ty: 1.25);
    g2.rotate( Math.toRadians(75));
    g2.scale(  sx: 0.1, sy: 2);
    filledRect(g2);
    g2.rotate( Math.toRadians(75));
    g2.setColor(saveColor);
    g2.setTransform(saveTransform);

}
private void bar2(Graphics2D g2) { // (DELETE THIS EXAMPLE)
    AffineTransform saveTransform = g2.getTransform();  // (It might be ne
    Color saveColor = g2.getColor();
    g2.setColor( Color.red);
    g2.translate( tx: 0, ty: -0.5);
    g2.rotate( Math.toRadians( 75 ));
    g2.scale(  sx: 0.2, sy: 2.5);
    filledRect(g2);
    g2.setColor(saveColor);
    g2.setTransform(saveTransform);
}
private void bar3(Graphics2D g2) { // (DELETE THIS EXAMPLE)
    AffineTransform saveTransform = g2.getTransform();  // (It might be ne
    Color saveColor = g2.getColor();
    g2.setColor( Color.red);
    g2.translate( tx: 2.5, ty: 1.3);
    g2.rotate( Math.toRadians(75));
    g2.scale(  sx: 0.09, sy: 1.5);
    filledRect(g2);
    g2.setColor(saveColor);
    g2.setTransform(saveTransform);
}
```

```java
private void triangle1(Graphics2D g2) { // (DELETE THIS EXAMPLE)
    AffineTransform saveTransform = g2.getTransform();  // (It might be necessary to save/restore tran
    Color saveColor = g2.getColor();
    g2.setColor( Color.magenta );
    g2.translate( tx: -2.5, ty: -0.5);
    g2.scale(  sx: 0.3, sy: 1.7);
    filledTriangle(g2);
    g2.setColor(saveColor);
    g2.setTransform(saveTransform);
}
private void triangle2(Graphics2D g2) { // (DELETE THIS EXAMPLE)
    AffineTransform saveTransform = g2.getTransform();  // (It might be necessary to save/restore tran
    Color saveColor = g2.getColor();
    g2.setColor( Color.blue );
    g2.translate( tx: 0, ty: -2.5);
    g2.scale(  sx: 0.5, sy: 2 );
    filledTriangle(g2);
    g2.setColor(saveColor);
    g2.setTransform(saveTransform);

}
private void triangle3(Graphics2D g2) { // (DELETE THIS EXAMPLE)
    AffineTransform saveTransform = g2.getTransform();  // (It might be necessary to save/restore tran
    Color saveColor = g2.getColor();
    g2.setColor( Color.green );
    g2.translate( tx: 2.5, ty: 0);
    g2.scale(  sx: 0.2, sy: 1.3 );
    filledTriangle(g2);
    g2.setColor(saveColor);
    g2.setTransform(saveTransform);
}
```

```java
private static void nShape(Graphics2D g2) {
    int N = 11;
    double[] xPoints = new double[N];
    double[] yPoints = new double[N];
    for(int i = 1 ; i<=N;i++)
    {
        xPoints[i-1]= (double) (1*Math.sin((2*Math.PI/N)*i));
    }
    for(int i = 1 ; i<=N;i++)
    {
        yPoints[i-1]= (double) (1*Math.cos((2*Math.PI/N)*i));
    }
    Path2D path = new Path2D.Double();
    path.moveTo(xPoints[0],yPoints[0]);
    for(int i = 0 ; i<N;i++)
    {
        path.lineTo(xPoints[i],yPoints[i]);
        path.moveTo( x: 0, y: 0);
        path.lineTo(xPoints[i],yPoints[i]);
    }
    path.lineTo(xPoints[0],yPoints[0]);
    path.closePath();
    g2.draw(path);
}
```

Zadanie 2. Graf

```
rotatingNshape1 = new TransformedObject(nShape);
rotatingNshape1.setTranslation( dx: -3.45, dy: 1.50).setScale( sx: 0.5, sy: 0.5).setColor(Color.black);
world.add(rotatingNshape1);
rotatingNshape2 = new TransformedObject(nShape);
rotatingNshape2.setTranslation( dx: -1.55, dy: 1).setScale( sx: 0.5, sy: 0.5).setColor(Color.black);
world.add(rotatingNshape2);
rotatingNshape3 = new TransformedObject(nShape);
rotatingNshape3.setTranslation( dx: -1.15, dy: -0.20).setScale( sx: 0.7, sy: 0.7).setColor(Color.black);
world.add(rotatingNshape3);
rotatingNshape4 = new TransformedObject(nShape);
rotatingNshape4.setTranslation( dx: 1.15, dy: -0.80).setScale( sx: 0.7, sy: 0.7).setColor(Color.black);
world.add(rotatingNshape4);
rotatingNshape5 = new TransformedObject(nShape);
rotatingNshape5.setTranslation( dx: 1.8, dy: 1.5).setScale( sx: 0.3, sy: 0.3).setColor(Color.black);
world.add(rotatingNshape5);
rotatingNshape6 = new TransformedObject(nShape);
rotatingNshape6.setTranslation( dx: 3.2, dy: 1.1).setScale( sx: 0.3, sy: 0.3).setColor(Color.black);
world.add(rotatingNshape6);
rectangle = new TransformedObject(filledRect); //bar1
rectangle.setTranslation( dx: -2.5, dy: 1.25).setScale( sx: 0.1, sy: 2).setRotation(75).setColor(Color.red);
world.add(rectangle);
rectangle = new TransformedObject(filledRect); //bar2
rectangle.setTranslation( dx: 0, dy: -0.5).setScale( sx: 0.2, sy: 2.5).setRotation(75).setColor(Color.red);
world.add(rectangle);
rectangle = new TransformedObject(filledRect);  //bar3
rectangle.setTranslation( dx: 2.5, dy: 1.3).setScale( sx: 0.09, sy: 1.5).setRotation(75).setColor(Color.red);
world.add(rectangle);
triangle = new TransformedObject(filledTriangle); //triangle1
triangle.setTranslation( dx: -2.5, dy: -0.5).setScale( sx: 0.3, sy: 1.7).setColor(Color.magenta);
world.add(triangle);
triangle = new TransformedObject(filledTriangle); //triangle2
triangle.setTranslation( dx: 0, dy: -2.5).setScale( sx: 0.5, sy: 2).setColor(Color.blue);
world.add(triangle);
triangle = new TransformedObject(filledTriangle); //triangle3
triangle.setTranslation( dx: 2.5, dy: 0).setScale( sx: 0.2, sy: 1.3).setColor(Color.green);
world.add(triangle);
```

```
private TransformedObject rotatingNshape1;
private TransformedObject rotatingNshape2;
private TransformedObject rotatingNshape3;
private TransformedObject rotatingNshape4;
private TransformedObject rotatingNshape5;
private TransformedObject rotatingNshape6;
private TransformedObject triangle;
private TransformedObject rectangle;
// (DELETE THIS EXAMPLE)
```

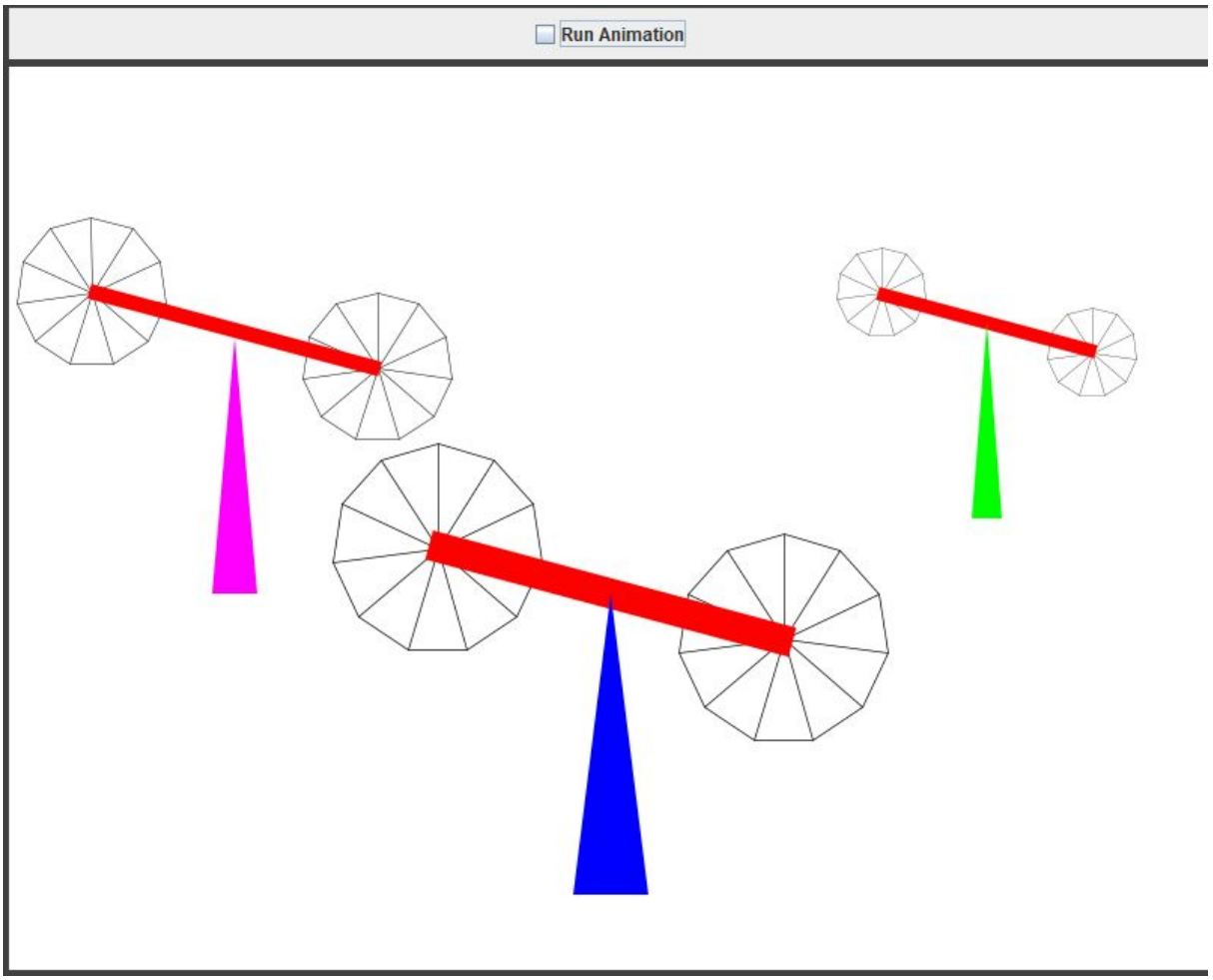
```java
public void updateFrame() {
    frameNumber++;
    // TODO: Update state in preparation for drawing the next frame.
    rotatingNshape1.setRotation(frameNumber);
    rotatingNshape2.setRotation(frameNumber);
    rotatingNshape3.setRotation(frameNumber);
    rotatingNshape4.setRotation(frameNumber);
    rotatingNshape5.setRotation(frameNumber);
    rotatingNshape6.setRotation(frameNumber);

}
```

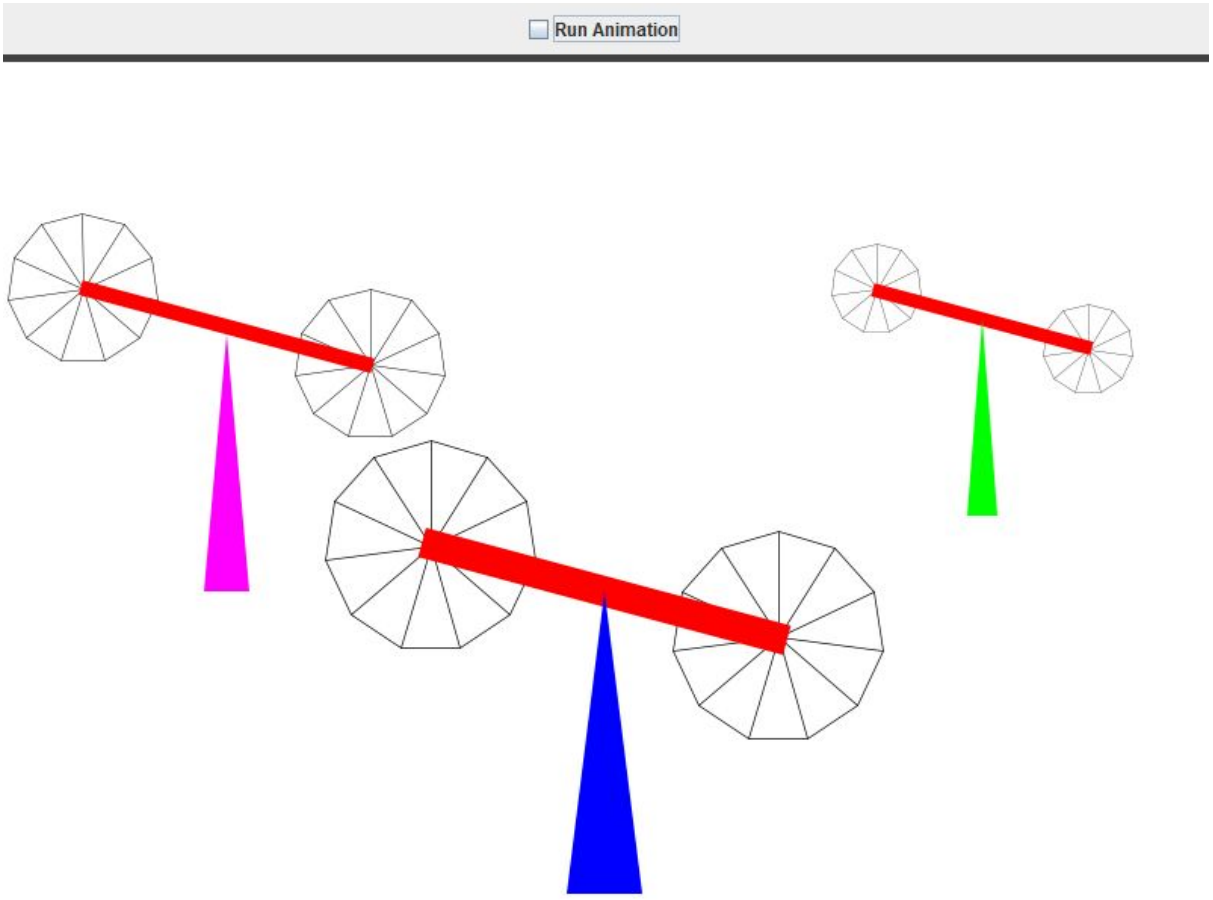
```java
private static SceneGraphNode nShape = (g) -> {
        double[] xPoints = new double[11];
        double[] yPoints = new double[11];
        for(int i = 1 ; i<=11;i++)
        {
            xPoints[i-1]= (double) (1*Math.sin((2*Math.PI/11)*i));
        }
        for(int i = 1 ; i<=11;i++)
        {
            yPoints[i-1]= (double) (1*Math.cos((2*Math.PI/11)*i));
        }
        Path2D path = new Path2D.Double();
        path.moveTo(xPoints[0],yPoints[0]);
        path.lineTo( x: 0, y: 0);
        path.moveTo(xPoints[0],yPoints[0]);
        for(int i = 1 ; i<11;i++)
        {
            path.lineTo(xPoints[i],yPoints[i]);
            path.moveTo( x: 0, y: 0);
            path.lineTo(xPoints[i],yPoints[i]);
        }
        path.lineTo(xPoints[0],yPoints[0]);
        path.closePath();
        g.draw(path);
};
```

Wyniki działania:

GRAF:



HIERARCHIA:



Podsumowanie:

Na podstawie otrzymanego wyniku można stwierdzić, że używając funkcji hierarchicznej, to możemy skorzystać z jednej funkcji wielokrotnie, musi posiadać ona wiele argumentów wejściowych. Z kolei korzystając ze sposobu obiektowego możemy dany obiekt transformować przy pomocy rozszerzonych metod.